All about CoAP #2

TP#4 using FIT/IoT-Lab

Lecture slides for RIO201

31-10-2018

For people who haven’t finished last week

- **Focus on that first!**
  - You need last week’s TP for your final project
- **This week**
  - Is more for your better understanding of sensor networks

Client on the sensor

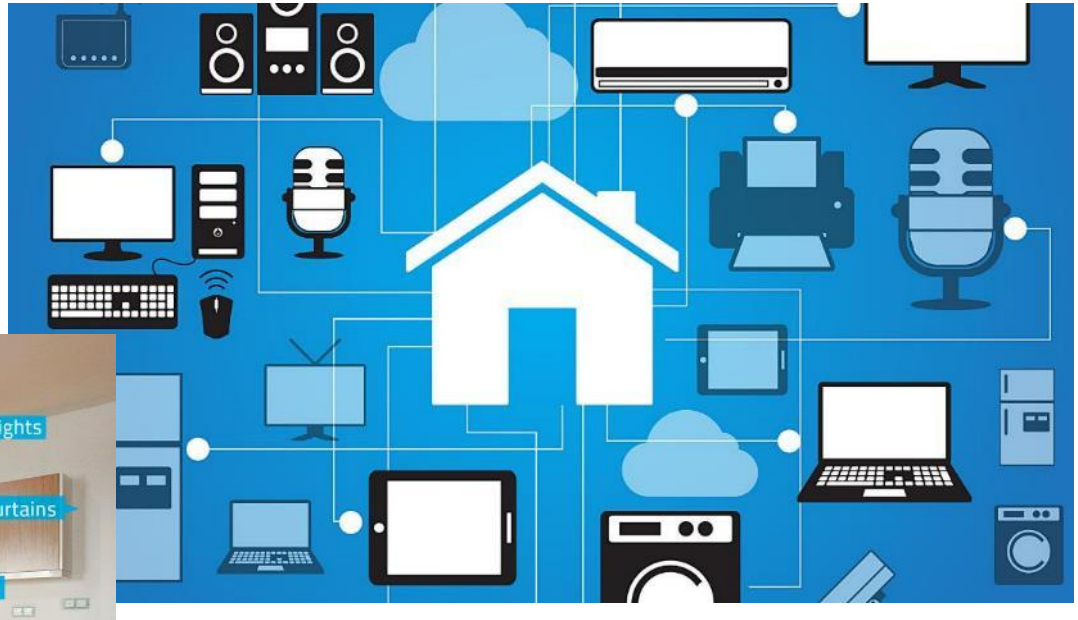
Up to now...

- **We have the CoAP clients only at the user's point of control**
  - Front-end
  - Local computers

- **But what if we want the sensors to talk with each other directly?**
  - Or rather, sensor ←→ actuator?
  - Actuator = CoAP client?
  - Sensor = CoAP server?

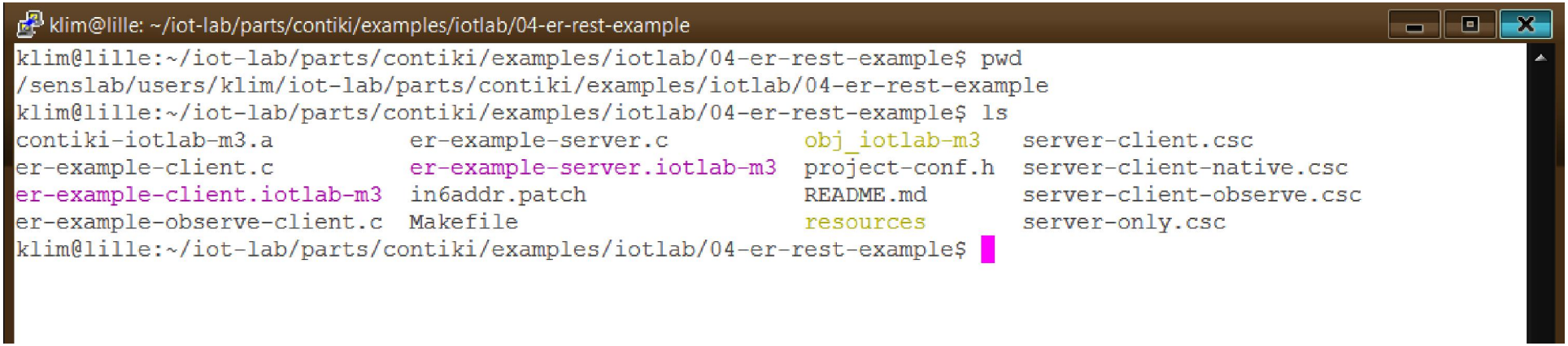
Why?

- **Do you remember the Smart Home environment?**



CoAP client for sensors

- **Goto**
  - ~/iot-lab/parts/contiki/examples/iotlab/04-er-rest-example
  - Nano er-example-client.c



CoAP client code

- https://github.com/contiki-os/contiki/blob/master/examples/er-rest-example/er-example-client.c

- **Focus on:**
  - Its basic operation (time period)
  - What is it doing? (toggle through post)
  - Sending and receiving

Let's change the code a bit!

- Debug code 0 → 1

```
#include "dev/button-sensor.h"

#define DEBUG 1
#if DEBUG
#include <stdio.h>
```

- URL for querying the server

```
/* Example URIs that can be queried. */
#define NUMBER_OF_URLS 4
/* leading and ending slashes only for demo purposes, get cropped automatically$
char *service_urls[NUMBER_OF_URLS] =
{ ".well-known/core", "/actuators/toggle", "battery/", "sensors/light" };
#if PLATFORM_HAS_BUTTON
```

Let's change the code a bit!

- **Type of request made**

```
/* prepare request, TID is set by COAP_BLOCKING_REQUEST() */
coap_init_message(request, COAP_TYPE_CON, COAP_GET, 0);
coap_set_header_uri_path(request, service_urls[3]);

//const char msg[] = " ";
PRINTF("%.*s\n",request);
```

- COAP_POST → COAP_GET
- Service_urls[1] → [3]

Don’t forget the address!

- Change the address to your CoAP server’s

```
 GNU nano 2.2.6                    Fichier : er-example-client.c

#define PRINT6ADDR(addr)
#define PRINTLLADDR(addr)
#endif

/* FIXME: This server address is hard-coded for Cooja and link-local for unconnected border router. */
#define SERVER_NODE(ipaddr)   uip_ip6addr(ipaddr, 0xfe80, 0, 0, 0, 0x0212, 0x7402, 0x0002, 0x0202)      /* coo
/* #define SERVER_NODE(ipaddr)   uip_ip6addr(ipaddr, 0xbbbb, 0, 0, 0, 0, 0, 0, 0x1) */

#define LOCAL_PORT      UIP_HTONS(COAP_DEFAULT_PORT + 1)
#define REMOTE_PORT     UIP_HTONS(COAP_DEFAULT_PORT)

#define TOGGLE_INTERVAL 10

PROCESS(er_example_client, "Erbium Example Client");
AUTOSTART_PROCESSES(&er_example_client);

uip_ipaddr_t server_ipaddr;
static struct etimer et;
```

Debug the CoAP engine

- **Let's try to print some information on the screens**
  - ~/iot-lab/parts/contiki/apps/er-coap
  - nano er-coap-engine.c

- **Change debug code from 0 to 1**

Recompile

- **Go back to the 04-er-rest-example**
  - Make TARGET=iotlab-m3
  - iotlab-node –update .....
  - Nc m3-XXX 20000 (do it for both client and server)

Exercise

- Change the interval of the actuator sending requests to the CoAP server
- Also, make several requests for different information (light, accel) to gather more information
- Finally, when receiving light information, if the light value changes, printf an alarm on the screen.

Ultimate challenge for today

- **If a CoAP client actuator generates an alarm,**
  - Right now we can only observe it through the nc command
- **DELIVER the alarm from the CoAP client to the user front-end**
  - If you are using python, to the python code
  - But how?